

Feasibility Based-Large Margin Nearest Neighbor Metric Learning

Babak Hosseini , Barbara Hammer

bhosseini, bhammer@techfak.uni-bielefeld.de

CITEC centre of excellence, Bielefeld University, Germany

This is preprint of a submitted conference paper as provided by the authors

Abstract

In the area of data classification, one of the prominent algorithms is the large margin nearest neighbor (LMNN) approach which is a metric learning to enhance the performance of the popular k-nearest neighbor classifier. In principles, LMNN learns a more efficient metric in the input space by using a linear mapping as the outcome of a convex optimization problem. However, one of the greatest weak points of LMNN is the strong reliance of its optimization paradigm on how the neighboring points are chosen. In this paper, it is mathematically proved for the first time that the regular way of choosing the target points can lead to non-feasible optimization conditions regardless of the number of chosen neighboring points. We present a mathematical approach to categorize the target points into feasible and infeasible exemplars, and also we provide a feasibility measure for preference of the target candidates. In our proposed Feasibility Based-LMNN algorithm, we use the above clue to construct the optimization problem based on the most promising general mapping directions in the input space. Our empirical results shows that via using the proposed FB-LMNN approach the optimization problem will converge in a better optimum point, and therefor leads to better classification on the well-known benchmark datasets

Keywords: Large margin nearest neighbor, Feasibility measure, Convex Optimization, K-nearest neighbor classifie.

1 Introduction

Metric learning as widely used approach in the area of discriminative data mining is related to finding a suitable metric for the given data in order to enhance the processing of specific dataset and mostly to improve the classification accuracy. In basic terms it tries to make similar class data closer and distance of them to other classes farther. One of the well-known methods for metric learning is the large margin nearest neighbor (LMNN) which is fundamentally destined to increase the performance of k-nearest neighbor classifier [1, 2, 3], by transferring the maximum margin concept of SVM classifier to the kNN framework. LMNN

was used in many real problems such as face recognition [4], motion classification [5] and pedestrian identification [6]. Furthermore, this adaptation of metric in this manner also enhances the model interpretability along with increasing the accuracy, and in some domain it is used for easier visualization of data [7, 8].

There has been several attempts to improve the performance of the original LMNN approach, such as complexity reduction of the optimization framework [9], eigenvalue based optimization [10], its extension to the multi-tasking problem [11] and a hierarchical preprocessing of data [12] all of which have interesting results. However there are still some open issues in the fundamental parts of this metric learning algorithm which hasn't been addressed yet.

To our knowledge, no one yet studied the idea of selecting the target points in a way rather than the sorted order of distances. One of the weak points of this approach is the fact that it always includes the nearest neighbors based on the current metric even if the size of the neighborhood is increased. And we believe this fact can decrease the algorithm performance in specific situations. Based on that perspective, in this paper we introduced a new insight into the optimization algorithm of LMNN approach and proposed a novel technique in order to enhance the selection process of neighborhood points based on some feasibility criteria. In the next sections we will explain the main concept of LMNN, then we talk about the feasibility measure to that problem and at the end we evaluate our method on real and artificial datasets.

2 Large Margin Nearest Neighbor Algorithm

Large margin nearest neighbor algorithm is a metric learning algorithm which is design to enhance the performance of nearest neighbor classifiers such as k-nearest neighbors (kNN). For set of labeled data vectors $\vec{x}_i \in \mathbb{R}^n$, $i = 1, \dots, m$ LMNN tries to find a quadratic metric which can increase the accuracy of kNN classifier. As the performance of kNN is highly depended on the distance of data points to their nearby neighbors, LMNN learns an optimized metric as

$$\mathcal{D}(\vec{x}_i, \vec{x}_j) = (\mathbf{L}(\vec{x}_i - \vec{x}_j))^2 = (\vec{x}_i - \vec{x}_j)^\top \mathbf{L}^\top \mathbf{L} (\vec{x}_i - \vec{x}_j) \quad (1)$$

in order to improve the neighborhood structure of data in a robust way. From another aspect of view, matrix \mathbf{L} is a linear mapping with the quadratic form of $\mathbf{M} := \mathbf{L}^\top \mathbf{L}$. LMNN algorithm's objective function is formed based on the k-nearest data samples \vec{x}_j (targets) to each data points \vec{x}_i having the same class labels. These sets of data which we note them as \mathcal{N}_i^k will be encoded in the optimization problem using $\eta_{ij} = 1$ in case $\vec{x}_j \in \mathcal{N}_i^k$. The cost function of LMNN is formed based on minimizing the distance to the nearby same class data samples and pushing away data samples from data point of other classes which are known as imposters:

$$\epsilon(\mathbf{L}) := \sum_{ij} \eta_{ij} \mathcal{D}(\vec{x}_i, \vec{x}_j) + m \sum_{ijl} \eta_{ij} (1 - \delta_{il}) \cdot [c + \mathcal{D}(\vec{x}_i, \vec{x}_j) - \mathcal{D}(\vec{x}_i, \vec{x}_l)]_+ \quad (2)$$

in which $[\cdot]_+$ is the Hinge loss function and $c > 0$ is a margin parameter, so that in the ideal case, all the imposters are pushed behind the margin of nearest

same label neighbors. As a notation, we consider the set of imposters points for \vec{x}_i with target neighbor \vec{x}_j as

$$\mathcal{I}m_i^j = \{\vec{x}_l : \mathcal{D}(\vec{x}_i, \vec{x}_l) < c + \mathcal{D}(\vec{x}_i, \vec{x}_j), c > 0, \vec{x}_j \in \mathcal{N}_i^k\}$$

There is another meta parameter in (2) as m which decides between the ratio of pull and push forces in the algorithm. According to [3] the optimization problem is a category of semidefinite programming as:

$$\begin{aligned} \min \quad & \sum_{ij} \mathcal{D}(\vec{x}_i, \vec{x}_j) + m \sum_{ijl} \eta_{ij} (1 - \delta_{il}) \xi_{ijl} \\ \text{where} \quad & \mathcal{D}(\vec{x}_i, \vec{x}_l) - \mathcal{D}(\vec{x}_i, \vec{x}_j) \geq c - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \\ & \mathbf{M} \succeq 0 \end{aligned} \quad (3)$$

the parameter $\delta_{il} = 1$ if i, l have same labels, so the second part of the optimization function will be considered only for \vec{x}_l that have different labels than \vec{x}_i . Based on the convexity of the problem, it is possible to optimize it with respect to a positive semidefinite matrix \mathbf{M} .

3 Feasibility of target neighborhoods

As explained in section 2 the first step in the LMNN approach is the selection of target neighbors \vec{x}_j for each input \vec{x}_i prior to solving the optimization problem (3). And the convexity of this optimization problem promises its convergence to a global minimum point M^* for the objective function; However the optimality of this point is respect to the selected set of all chosen target points $\mathcal{N}^k = \{\mathcal{N}_i^k, \forall i = 1, \dots, N\}$ which is done based on the initial metric (L). Therefore, in many real problems it is possible to find a different set \mathbf{S}' based on which the optimization problem can reach another optimum point M' which leads to a lower objective value. In other words, the global optimum point to (3) can be a local optimum point for the general optimization problem

$$\begin{aligned} \min_{M, \eta} \quad & \sum_{ij} \eta_{ij} \mathcal{D}(\vec{x}_i, \vec{x}_j) + m \sum_{ijl} \eta_{ij} (1 - \delta_{il}) \xi_{ijl} \\ \text{s.t.} \quad & \mathcal{D}(\vec{x}_i, \vec{x}_l) - \mathcal{D}(\vec{x}_i, \vec{x}_j) \geq c - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \\ & M \succeq 0 \end{aligned} \quad (4)$$

which takes into account also the parameter η or in other words, considering different possibilities for the target neighbors \mathcal{N}_i^k for a given k . The optimization problem (4) is considered a NP-hard problem Which is computationally difficult to solve. One way to get close to the global optimum of problem is to use the multiple pass technique suggested by [3], which tries to update the targets neighbors \mathcal{N}^k based on the learned metric L after the convergence of the optimization problem (3). Then repeat the optimization again based on the updated η_{ij} and the last value of L , and repeat these iterations until there will be no further change in \mathcal{N}^k . It is proved in [13] that in each iteration t of multiple pass LMNN, the optimization problem will converge to a better or equal optimal point L^t comparing to L^t . This technique tries to push the algorithm towards a the global optimum direction, however it cannot promise the

convergence to the global optimum because it is still depended to the initial and following choices of \mathcal{N}^k in the optimization process.

Furthermore, There is another major weak-point in the main principles of LMNN which to our knowledge has not been addressed in the literature by anyone else. Based on the given backgrounds in section 2, the algorithm selects the target neighbors $\mathbf{S} = \{\mathcal{N}_i^k\}$ in the order of their distance from \vec{x}_i . Meaning that the $\mathcal{N}_i^1 \subset \mathcal{N}_i^2 \subset \dots \subset \mathcal{N}_i^k$ which we see it as a general restriction in the optimization problem. Relevantly, It can be directly concluded from the (4) that solving the main optimization problem needs having more flexibility in choosing the target neighbors.

For example, consider a sample arrangement of data points in a 2-D space as in Fig. 1 in which \vec{x}_i is the main data point and $\{\vec{x}_j, j = 1, \dots, 3\}$ are the candidate targets and \vec{x}_4 is a nearby data from other classes which are called imposters. As you can see, if we choose \vec{x}_1 as the target point for \vec{x}_i based on their small distance, consequently \vec{x}_4 will be selected as the relevant imposter and the algorithm tries to send \vec{x}_4 beyond \vec{x}_i using a linear mapping L . At the same time due to the first part of the objective function in (3), it tries to bring $\{\vec{x}_j, j = 1, \dots, 3\}$ closer to \vec{x}_i . However as the triple (i,1,4) are positioned on a straight line, they will still remain on a line after applying any linear transform L , and also the ratio of their distances will be preserved, because L is a non-singular affine transform. Or in mathematical terms, because $(\vec{x}_1 - \vec{x}_i) = \alpha(\vec{x}_4 - \vec{x}_i)$, $\alpha > 1$, under the linear transform L we will have $(L\vec{x}_1 - L\vec{x}_i) = \alpha(L\vec{x}_4 - L\vec{x}_i)$ or $(\vec{x}'_1 - \vec{x}'_i) = \alpha(\vec{x}'_4 - \vec{x}'_i)$ which means the new points are in the same relational positions as before (1.b). Even through out increasing the neighborhood size k and adding more targets like \vec{x}_2 and \vec{x}_3 to \mathcal{N}_i^k , the optimization problem still tries to bring \vec{x}_1 closer to \vec{x}_i which inevitably brings the imposter \vec{x}_4 closer too, and ending in a weak optimum point in the scope of (4). In more general terms, neighboring points like \vec{x}_1 are illegal targets for \vec{x}_i due to the infeasibility of the relevant triples, and having such infeasible triples in our optimization framework will mislead the algorithm to local minimums in the scope of the main optimization problem. Although the algorithm might reduce the objective function by bringing same class data closer to each others like Fig. 1.b, but based on kNN classifier principles \vec{x}_4 will be wrongly chosen as the same label sample for the test data \vec{x}_i .

In order to address this problem mathematically we will refer to the relaxation part in (3) which tries to make

$$\mathcal{D}(\vec{x}_i, \vec{x}_j) + c \leq \mathcal{D}(\vec{x}_i, \vec{x}_l) \quad (5)$$

however in cases like the example in Fig. 1 for the triples $(i, 1, 4)$ the inequality (5) is not feasible for any linear mapping L as the solution. Hence, we are interested in defining a criteria in order to mathematically check the feasibility of the above inequality.

Theorem 1 *A triple set of (i, j, l) will be infeasible for the LMNN optimization \iff the angle between $(\vec{x}_i - \vec{x}_j)$ and $(\vec{x}_i - \vec{x}_l)$ is zero.*

Proof 1 *Assuming that $(\vec{x}_i - \vec{x}_j) > (\vec{x}_i - \vec{x}_l)$ and we want (5) to be feasible. As by the definition, $c \geq 0$, so it is required to have $\mathcal{D}(\vec{x}_i, \vec{x}_j) \leq \mathcal{D}(\vec{x}_i, \vec{x}_l)$, which means*

$$(\vec{x}_i - \vec{x}_j)^T M(\vec{x}_i - \vec{x}_j) - (\vec{x}_i - \vec{x}_l)^T M(\vec{x}_i - \vec{x}_l) \leq 0$$

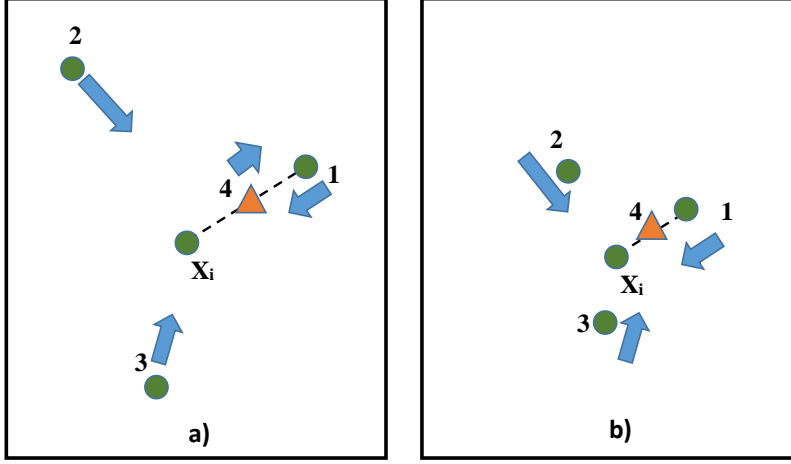


Figure 1: A 2-D example of a triple of data for which the optimization constraint is infeasible

or

$$\text{Tr}\{((\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^T - (\vec{x}_i - \vec{x}_l)(\vec{x}_i - \vec{x}_l)^T)M\} \leq 0 \quad (6)$$

let's call it $\text{Tr}\{QM\} \leq 0$. Based on the semidefinite structure of the optimization problem, M is a positive semidefinite matrix. Then if matrix Q happens to be positive semidefinite too, the resulting matrix QM will be positive semidefinite as well as having positive eigenvalues. Therefore, $\text{Tr}\{QM\} > 0$ which means that (5) becomes infeasible.

For simplicity we consider $Q = \vec{a}\vec{a}^T - \vec{b}\vec{b}^T$, meaning that $Q \in \text{span}\{\vec{a}, \vec{b}\}$ and follows the fact that Q has at most two non-zero eigenvalues. If we assume \vec{a}, \vec{b} linearly independent, then the restriction matrix of Q to $U = \text{span}\{\vec{a}, \vec{b}\}$ would be

$$\begin{bmatrix} \vec{a} \cdot \vec{a} & \vec{a} \cdot \vec{b} \\ -\vec{a} \cdot \vec{b} & -\vec{b} \cdot \vec{a} \end{bmatrix}$$

and its determinant or the multiplication the eigenvalues would be

$$-\|\vec{a}\|^2\|\vec{b}\|^2 + (\vec{a} \cdot \vec{b})^2 = -\|\vec{a}\|^2\|\vec{b}\|^2 \sin^2 \theta$$

in which θ is the angle between the two vectors \vec{a} and \vec{b} . The determinant is always negative, unless $\theta = 0$, which in that case $Q = (\alpha^2 - 1)\vec{b}\vec{b}^T$ as $\vec{a} = \alpha\vec{b}, \alpha > 1$ by the first assumption. As a result Q would be a positive semidefinite matrix with one positive eigenvalue accordingly; Consequently (6) and (5) become infeasible.

Based on the above demonstrations and the support of theorem 1, it is possible that exist infeasible target set

$$\mathcal{F}_i^k = \{\vec{x}_j : \vec{x}_j \in \mathcal{N}_i^k, \mathcal{D}(\vec{x}_i, \vec{x}_j) > \mathcal{D}(\vec{x}_i, \vec{x}_l), \forall M \succeq 0\}$$

in the k vicinity of \vec{x}_i which can influence the optimization problem dramatically. Another important fact is that with normal selection of target neighbors \mathcal{N}_i^k based on their sorted distance to \vec{x}_i , infeasible points $\mathcal{F}_i^k \in \mathcal{N}_i^{k'}$ for any $k' > k$, so increasing the neighborhood size cannot guarantee to eliminate the role of infeasible targets.

In the next section we will present an effective approach to benefit from proposed concept of feasibility check in order to enhance the optimization algorithm.

4 Feasibility Based Large Margin Nearest Neighbor

Based on the discussion in the previous section, \mathcal{F}^k is set of (\vec{x}_i, \vec{x}_j) which are infeasible based on the optimization constraint and we see them as the singular conditions for the optimization algorithm. One way to prevent the infeasible target set \mathcal{F}^k from influencing the optimization problem (4) is to simply rule them out prior to the start of the optimization algorithm as a pre-processing step. To do so, we check the relevant nearby imposters \vec{x}_l for each pair of (\vec{x}_i, \vec{x}_j) and in case of existing any infeasible triples in combination with (\vec{x}_i, \vec{x}_j) will be eliminated from \mathcal{N}_i^k by making $\eta_{ij} = 0$ in (3) otherwise $W_{ij} = 1$. Then we search for the next closest neighbor to \vec{x}_i in order to substitute \vec{x}_j with that neighbor.

4.1 Feasibility Bounds.

In principles, that strategy is effective against the infeasible cases like in the imaginary situation in Fig. 1, however in real experiments the chances to observe a definite cases of infeasibility might be small, for example the smallest eigenvalue of Q in theorem 1 can be a small ϵ but not equal to zero. Nevertheless we will show the introduced feasibility concept plays a direct rule in the performance of LMNN algorithm and definitely has to be considered in the optimization algorithm. For example consider a more realistic 2-D situation like Fig. 2 in which the \vec{x}_4 is an imposter placed between \vec{x}_i and \vec{x}_1 and so close to their connecting line, however according to theorem 1 the smallest eigenvalue of matrix Q has very small positive value. Hence, it is still feasible to find a linear mapping L to make (5) feasible like in Fig. 2.a. Although this mapping makes the cost function smaller and the hing loss equal to zero for the data triple (i,1,4), but it is so tight due the small eigenvalue of Q in one of the directions. In other words, in order to pull \vec{x}_1 toward \vec{x}_i and to pull \vec{x}_l away, the algorithm is bounded to use a relevant set S of possible metrics with restricted shapes

$$S_{ijl} = \{M : M \succeq 0, \mathcal{D}(\vec{x}_i, \vec{x}_j) < \mathcal{D}(\vec{x}_i, \vec{x}_l)\} \quad (7)$$

And if it tries to find an optimal $S^* \in S_{i,1,4}$ to apply to other data points in the problem, the probability of its failure can be high Fig. 2.b. In comparison, if we take \vec{x}_2 as the target to \vec{x}_i , then based on the wide angle between $(\vec{x}_4 - \vec{x}_i)$ and $(\vec{x}_2 - \vec{x}_i)$, the eigenvalues of relevant matrix Q will be relatively bigger than the previous case, and easily can be inferred from the Fig. 2 that $S_{i,4,2}$

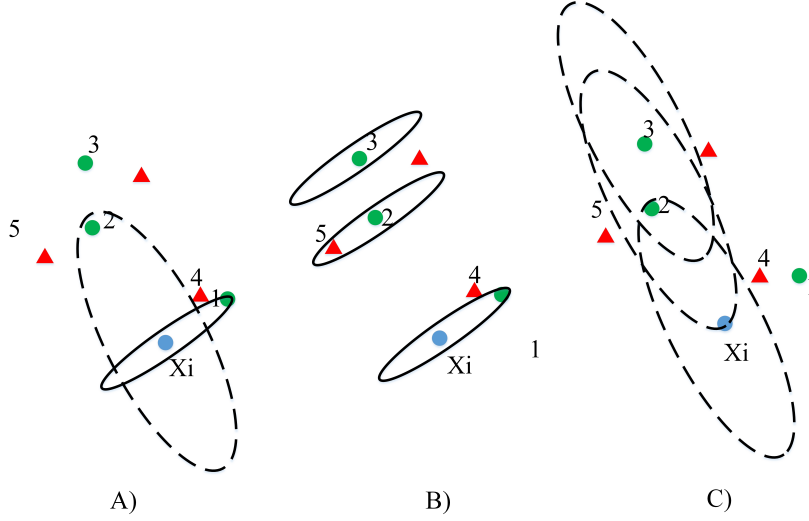


Figure 2: A 2-D example in which the tight set of feasible solutions for triple (i,1,4) fails for other data samples, however a wider set considering the target neighbor \vec{x}_2 has a higher chance to catch the nearby same class neighbors and avoid imposters

includes wider range of possible shapes which gives the optimization algorithm more flexibility to find a suitable metric $M^* \in S_{i,2,4}$ to fit to other data samples as in Fig. 2.c. As a result, for the sake of optimization (4), it would be better to put more efforts on optimizing the situation toward \vec{x}_2 than \vec{x}_1 .

In order to mathematically address this problem, we investigate the characteristics of feasible set S to the inequality (6), and more precisely we check the conditions in which the restrictions on the solution set S will be tighter or looser.

Lemma 1 For any Hermitian matrix Q and a positive semidefinite matrix M , $\lambda_k(Q)\lambda_{\min}(M) \leq \lambda_k(\text{Tr}(QM))$.

Proof 2 Because M is positive semidefinite, $\lambda_k(QM) = \lambda_k(Q\sqrt{M}\sqrt{M})$ where \sqrt{M} is the square root matrix of M . And since, Q and M are symmetric, $\lambda_k(QM) = \lambda_k(\sqrt{M}Q\sqrt{M})$. And according to the min-max theorem $\lambda_k(QM)$ is equal to

$$\begin{aligned} & \min_{\substack{F \subset \mathbb{R}^n \\ \dim(F)=k}} \left(\max_{x \in F \setminus \{0\}} \frac{(\sqrt{M}Q\sqrt{M}x, x)}{(x, x)} \right) \\ &= \min_{\substack{F \subset \mathbb{R}^n \\ \dim(F)=k}} \left(\max_{x \in F \setminus \{0\}} \frac{(Q\sqrt{M}x, \sqrt{M}x)}{(\sqrt{M}x, \sqrt{M}x)} \frac{(Mx, x)}{(x, x)} \right). \end{aligned}$$

As we know that $\lambda_{\min}(M) \leq \frac{\langle Mx, x \rangle}{\langle x, x \rangle} \leq \lambda_n(M)$, then

$$\lambda_k(QM) \geq \lambda_{\min}(M) \min_{\substack{F \subset \mathbb{R}^n \\ \dim(F)=k}} \left(\max_{x \in F \setminus \{0\}} \frac{(Q\sqrt{M}x, \sqrt{M}x)}{(\sqrt{M}x, \sqrt{M}x)} \right)$$

and based on the min-max theorem it means $\lambda_k(QM) \geq \lambda_{\min}(M)\lambda_k(Q)$.

Based on lemma 1 we have

$$\lambda_{\max}(Q)\lambda_{\min}(M) \leq \lambda_{\max}(QM). \quad (8)$$

Also it is proven in [14] that for our set of $\{Q, M\}$

$$\lambda_{\min}(Q)\lambda_{\max}(M) \leq \lambda_{\min}(QM)$$

adding the above to (8) will result in

$$\lambda_{\min}(Q)\lambda_{\max}(M) + \lambda_{\max}(Q)\lambda_{\min}(M) \leq \text{Tr}(QM)$$

As we want $\text{Tr}(QM) < 0$ to happen, the left part of the above inequality should be negative, meaning that

$$\frac{|\lambda_{\min}(Q)|}{\lambda_{\max}(Q)} > \frac{\lambda_{\min}(M)}{\lambda_{\max}(M)} \quad (9)$$

As the first clear conclusion, in case of $\lambda_{\min}(Q) = 0$ it results in the infeasibility of (9) which agrees with the outcome of theorem 1. The second important point about (9) is the fact that as the absolute value of $\lambda_{\min}(Q)$ gets smaller, the restriction on the values of $\lambda_{\min}(M)$ will be tighter which force the value of them to be smaller in size. Also another supporting fact to the above claim is proved in [15] as

$$\lambda_{\min}(Q) \text{Tr}(M) \leq \text{Tr}(QM)$$

Which imposes the bound on all eigenvalues of M so if $\lambda_{\min}(Q)$ becomes really small it forces $\|\lambda(M)\|_1$ to shrink in size. So although given any matrix Q with even really small eigenvalues the size of solution set S will be indefinite, but there would be strict bounds imposed on the shape (or boundaries of) of the solution set spectrahedron. In the following we are going to see how the above findings can improve the solution to the optimization problem (4)

4.2 Feasibility Bound Weights in the Optimization.

Considering the global solution to the LMNN optimization problem, we are looking for a metric transform L^* which can be the global optimum point to (4). So although the first part of the objective is necessary to help the optimization algorithm to converge to an optimum point, at the end, an ideal global optimum with respect to nearest neighbor classification would be a metric L^* , that fulfills (5) for all the triples inside set \mathcal{N}^k . In other words,

$$L^* \in \bigcap_{i,j \in \mathcal{N}^k, \forall l} \{S_{ijl}\}$$

according to (10). As result, if the problem has a linear solution for the kNN classifier, that intersection will not be empty. However in most of real problems, there won't be an intersection between all the solution sets, but still there is a best solution \hat{L}^* such that

$$\max_{\hat{L}^*} |S|, \{S : S = \bigcap_{i,j \in \mathcal{N}^k} \{S_{ijl}\}, S \neq \emptyset\} \quad (10)$$

Again we point out the important fact that this optimality is with respect to the nearest neighbor criteria used for prediction in a kNN classifier which highly depends on the distance order of nearby data samples than the value of their distances. Concluding based on (10), if the optimization problem (3) focuses more on data triples (i,j,l) with feasibility sets S_{ijl} that have wider restrictions on their eigenvalues, then the chances for learning a metric close to \hat{L}^* will be increased. According to (9), a good estimate for the tightness of the feasibility set S_{ijl} would be

$$\frac{|\lambda_{\min}(Q_{ijl})|}{\lambda_{\max}(Q_{ijl})} \quad (11)$$

As a result, we want to modify the convex optimization problem (3) with respect to the above measure. Furthermore, as we discuss before, we are also interested to know which neighboring points inside \mathcal{N}_i^k are more important or more promising to be considered as the important targets for \vec{x}_i . To do so, for each $l \in \mathcal{I}m_i^j$, we calculate (11) and assign $R_{ij} = \frac{|\lambda_{\min}(Q_{ijl^o})|}{\lambda_{\max}(Q_{ijl^o})}$ in which l^o is the \vec{x}_l that results in the smallest (11). In other words, $S_{ijl^o} = \bigcap_{l \in \mathcal{I}m_i^j} \{S_{ijl}\}$ which means the set of feasible solutions for the couple (i,j) , therefore we can take R_{ij} as a measure on how promising \vec{x}_j could be as a target for \vec{x}_i according to the optimization structure.

In order to adapt the optimization with respect to R , a greedy strategy would be to choose only targets neighbors for each \vec{x}_i with high R_{ij} values, and ruling out others from \mathcal{N}_i^k by making $\eta_{ij} = 0$ in the optimization problem (3). This will be despite the fact that they might be even the closets samples to \vec{x}_i based on the current metric L . However in order not to restrict the flexibility of the optimization algorithm we augment the weight matrix R in the optimization problem as (12). Doing so, the optimization algorithm puts more gains on the optimization cost of target neighbors which have solution sets with wider spectrahedrons.

$$\begin{aligned} \min_M \quad & \sum_{ij} \eta_{ij} R_{ij} \mathcal{D}(\vec{x}_i, \vec{x}_j) + m \sum_{ijl} \eta_{ij} R_{ij} (1 - \delta_{il}) \xi_{ijl} \\ \text{s.t.} \quad & \mathcal{D}(\vec{x}_i, \vec{x}_l) - \mathcal{D}(\vec{x}_i, \vec{x}_j) \geq c - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \\ & M \succeq 0 \end{aligned} \quad (12)$$

Another important fact about the weighted structure of (12) is that, it also considers the number of similar data points with respect to the condition of their solution sets. For example consider a data type in which the majority of the data samples are distributed such that the optimum metric L^* will be the $S' \subset S$ as the intersection of the tightest sets from S . In that situation, the

optimization algorithm is still able to choose those targets, because the general weight of the subset S' will becomes bigger through its repeated samples in the optimization framework. Also it is clear that considering the case of having complete infeasible target points will result in $R_{ij} = 0$ for $j \in \mathcal{F}_i^k$ and practically they will be removed from the optimization framework.

In order to implement the FB-LMNN algorithm, first as a pre-processing step the \mathcal{N}_i^k for each data point should be determined based on its k-nearest same-class neighbors using the original metric L (or current metric in the multi-pass loop). Then for each $j \in \mathcal{N}_i^k$ we find $\{l : \mathcal{D}(\vec{x}_i, \vec{x}_j) + c > \mathcal{D}(\vec{x}_i, \vec{x}_l)\}$ for which we calculate (11) and then R_{ij} will be determined. However, in practice we noticed that it is better to normalize $R_{i,:}$ for each \vec{x}_i which improve the scalability of R over all data samples. Having weight matrix R ready, we can start solving 12 as the second part of the Algorithm 1.

<p>Input: Metric \mathbf{M}, Data matrix $\mathbf{X} \in \mathbb{R}^{d \times N}$, k</p> <pre> 1 for $i = 1, \dots, N$ do 2 calculate \mathcal{N}_i^k; 3 for $j \in \mathcal{N}_i^k$ do 4 calculate $\mathcal{I}m_i^j$; 5 $R_{ij} = \min_l \frac{ \lambda_{min}(Q_{ijl}) }{\lambda_{max}(Q_{ijl})}, l \in \mathcal{I}m_i^j$ 6 end 7 $R_{i,:} = \frac{\sum_j R_{i,j}}{\max(R_{i,:})}$ 8 end 9 update R_{ij} in (12); 10 solve (12) using SD programming ;</pre>

Algorithm 1: Feasibility Based LMNN

5 Experiments

In this section we investigate the performance of the explained FB-LMNN algorithm. We implement our algorithm on some real data with different structures and also on a synthetic data to better show the effective point of the approach. We use the 10-fold cross validation to split the data into test and train batches. For choosing the parameter k as the number of nearest neighbors we used cross validation and we fix the parameter $m = 0$ which generally results in good performance for practical applications. The summary of datasets and the chosen meta parameter for the LMNN approach is brought in Table. 1 We compare our FB-LMNN algorithm first with kNN using Euclidean distance, and as the second approach we used the original LMNN method introduced by [3], also we tried the regular LMNN algorithm in a multi-pass fashion as [13] in order to improve its performance. As another rival linear classifier we checked our results also with multiclass SVM algorithm with RBF kernel.

5.1 Synthetic Data

To better demonstrate the effectiveness of the approach, we manually generated the so called zebra stripe toy data set which was also tried by the original and multi-pass versions of LMNN algorithm by [3] and [13]. As you can see in Fig. 3 according to the alternate distribution of data classes, the first suggestion to handle the task is to solve it by focusing on local distances between the data. Therefore nearest neighbor classifiers should be an ideal choice based on their principles. However the main difference of our dataset and the ones used in [3] and [13], is the ratio of the local distance to the nearest class data comparing to the distance to the nearest different class, similar to the example data in Fig. 1. based on the original metric, the closest target neighbor for each data sample \vec{x}_i belongs to a different row than \vec{x}_i row based on its smaller proximity in the horizontal axis. We take this characteristic to a more extreme case in which the distance to a same row target is so bigger than to a target in a different stripe, so the normal optimization problem (3) will be trapped in a local minimum (with respect to the global problem (4)) and even a multi-pass loop of the optimization will be trapped in a global path of optimization which will not get closer to the global optimum point. Fig. 4 shows the result of normal multi-pass LMNN at the convergence point. As you can see algorithm tried to scale and rotate the data in order to reduce the total cost function of optimization but was not able to solve the problem efficiently due to the tight feasibility sets of the nearest neighbor data triples. The accuracy of the multi-pass LMNN was 23% for this artificial data which was almost the same as the accuracy of vanilla kNN classifier. Although by increasing the number of k the good targets (same stripe data) can be selected too, but the wrong targets still will be a part of the optimization which prevent the algorithm to move on a optimum path.

On the other hand, the proposed FB-LMNN considers the feasibility of triples of the data. As a result, the triple sets in which the data point and the target are located on different rows are either infeasible or so narrow, while the feasible bounds will be wider for same row targets. As a result there will be bigger R_{ijl} weights for optimization the metric respect to latter set of targets which or in general to the vertical axis. Hence optimization problem leads to better optimum points as you see in Fig. 5, which shows that the algorithm focused on the data on the target neighbors on the same stripe even it was farther than the nearest neighbors. At the end it rotated and compressed the data along the stripes which resulted in 76% accuracy. Although this artificial data strongly supports benefit of considering feasibility sets for LMNN, we also provided implementation on real datasets to be confident about the performance of the approach in real situations.

5.2 Real Data

The real data sets are all chosen from the UCI repository library [16] and the results are reproducible easily. The selected data sets are related to different classification problems in different domain and also with different structures. The result of application of kNN, multi-pass LMNN, SVM and the Feasibility based LMNN are presented in Table 2. According to the results, for some datasets such as Car Evaluation and Iris the difference between regular LMNN

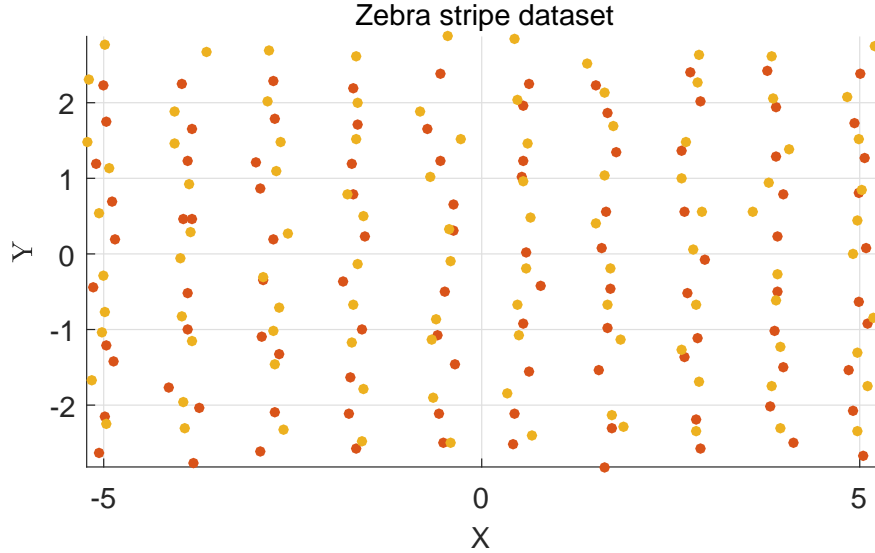


Figure 3: The zebra stripes data set as the synthetic dataset used for illustration. The same class data are distributed in each row and they classes rows are alternated for difficulty. There is added noise to the data as well

and the FB-LMNN is subtle, however in some other datasets the FB-LMNN has a higher performance than the Multi-pass LMNN showing that considering the form of feasibility sets for \mathcal{N}^k was effective, or in other words, there was similarity between same class data samples based on that feature. Comparing with SVM classifier, for few of the datasets SVM has a better result than the LMNN approaches which is based on the structure of the data which was more suitable for that category of classifiers.

6 Conclusion

In this paper, we addressed an important issue in the Large Margin metric learning algorithm for the first time, which is related to the distance based selection of optimization targets regardless of their importance or influence to the optimization problem. Even though the optimization problem of LMNN converges to a global minimum because of its convexity, that point is optimal with respect to the selected target Neighbors for the algorithm. We presented examples in which the typical way of changing neighborhood distance will not handle this problem. We stated that regardless of the distance of same class data, there might exist closed but infeasible neighboring points which can mislead the optimization path if we take them as target exemplars. We mathematically demonstrated how to detect and measure the infeasibility of the target points and rearrange the optimization problem to consider the above conditions during its process. As a result we proposed the FB-LMNN which is an alternating optimization optimization framework that tries to consider directions in the metric space in

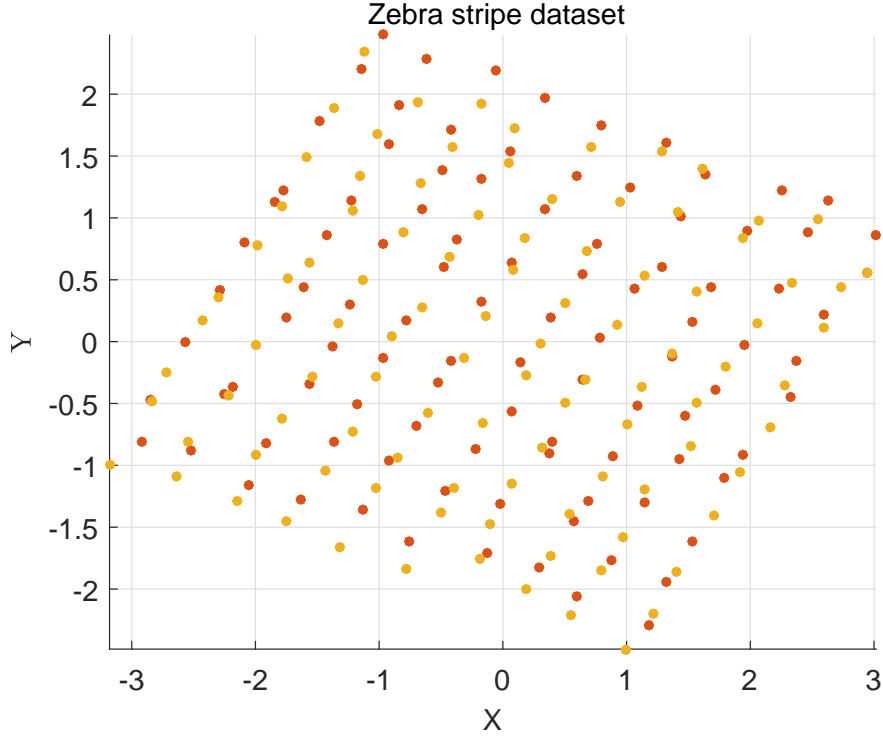


Figure 4: The result of projection of zebra dataset under the mapping learned by multi-pass LMNN

which there is a higher chance for an optimum convergence point. We presented our method with mathematical proves showing that the measure of feasibility of data triples has to be considered in the optimization problem. We illustrated the algorithms performance on artificial data set to show the strong point of the approach and also evaluated its performance for real problems, which showed the effectiveness of the proposed methodology. Although in some real data sets there is no difference between using the regular LMNN or the feasibility based, we still encourage using this extra measure as a safety approach to prevent from local optimum points. As a future work, the mentioned feasibility measure can be used to partition the data space into the clusters of data which are similar with respect to the direction of the optimal malahobian metric. We strongly believe that this is a promising starting point to have more detail insight into better selection of target points in LMNN as a metric adaptation algorithm which highly depends on the mentioned criteria.

7 Acknowledgment

This research was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the

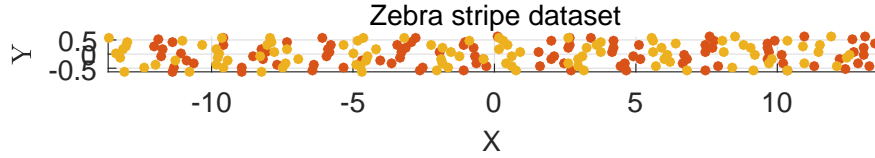


Figure 5: The result of projection of zebra dataset under the mapping learned by Feasibility Based LMNN

Table 1: The selected datasets and their information along with the chosen parameters for LMNN approach

Dataset	nC	nF	#data	Klmnn	kNN
Zebra	2	2	200	6	1
Wine	3	13	178	6	1
Balance	3	4	625	5	3
B. Cancer	20	30	256	5	3
Car Eval.	4	6	1728	9	3
Tic-Tac-Toe	2	9	958	3	1
Hepatitis	2	17	129	5	1
iris	3	4	128	6	1

German Research Foundation (DFG).

References

- [1] A. Bellet, A. Habrard, and M. Sebban, “A survey on metric learning for feature vectors and structured data,” *arXiv preprint arXiv:1306.6709*, 2013.
- [2] B. Kulis, “Metric learning: A survey,” *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.
- [3] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009. [Online]. Available: <http://doi.acm.org/10.1145/1577069.1577078>
- [4] M. Guillaumin, J. Verbeek, and C. Schmid, “Is that you? metric learning approaches for face identification,” in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 498–505.
- [5] B. Hosseini and B. Hammer, “Efficient metric learning for the analysis of motion data,” in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, Oct 2015, pp. 1–10.
- [6] M. Dikmen, E. Akbas, T. S. Huang, and N. Ahuja, “Pedestrian recognition with a learned metric,” in *Asian conference on Computer vision*. Springer, 2010, pp. 501–512.

Table 2: Classification accuracy(%) for the selected datasets and the chosen approaches

Dataset	kNN	MP-LMNN	FB-LMNN	SVM
Zebra	21.31	23.51	72.21	50.82
Wine	73.4	96.91	98.77	77.23
Balance	87.42	94.03	96.08	97.5
B. Cancer	94.66	96.68	97.07	78.49
Car Eval.	92.57	98.32	98.4	60.08
Tic-Tac-Toe	87.42	97.66	98.13	85
Hepatitis	84.16	84.46	90	79.11
iris	92.64	93.24	94.12	97.47

- [7] M. Biehl, K. Bunte, and P. Schneider, “Analysis of flow cytometry data by matrix relevance learning vector quantization,” *PLoS One*, vol. 8, no. 3, p. e59401, 2013.
- [8] A. Backhaus and U. Seiffert, “Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size,” *Neurocomputing*, vol. 131, pp. 15–22, 2014.
- [9] K. Park, C. Shen, Z. Hao, J. Kim *et al.*, “Efficiently learning a distance metric for large margin nearest neighbor classification,” in *AAAI*, 2011.
- [10] Y. Ying and P. Li, “Distance metric learning with eigenvalue optimization,” *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 1–26, 2012.
- [11] S. Parameswaran and K. Q. Weinberger, “Large margin multi-task metric learning,” in *Advances in neural information processing systems*, 2010, pp. 1867–1875.
- [12] Q. Chen and S. Sun, “Hierarchical large margin nearest neighbor classification,” in *Pattern Recognition (ICPR), 2010 20th International Conference on.* IEEE, 2010, pp. 906–909.
- [13] C. Göpfert, B. Paassen, and B. Hammer, “Convergence of multi-pass large margin nearest neighbor metric learning,” in *International Conference on Artificial Neural Networks.* Springer, 2016, pp. 510–517.
- [14] F. Zhang, Q. Zhang *et al.*, “Eigenvalue inequalities for matrix product,” *IEEE Transactions on Automatic Control*, vol. 51, no. 9, p. 1506, 2006.
- [15] S.-D. Wang, T.-S. Kuo, and C.-F. Hsu, “Trace bounds on the solution of the algebraic matrix riccati and lyapunov equation,” *IEEE Transactions on Automatic Control*, vol. 31, no. 7, pp. 654–656, 1986.
- [16] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: <http://archive.ics.uci.edu/ml>